RAPPORT – PROJET C

Le sujet consiste à affecter des familles à des jours de la semaine en respectant certaines contraintes, tout en minimisant une fonction de coût non linéaire. Nous avons certaines contraintes (nombres maximal d'individu par jour, nombre minimal d'individu par jour, toutes les familles doivent être toutes placées …) et l'objectif ici est de minimiser la fonction coût, qui dépend du choix du jour de la semaine qui leur est affecté. Ce coût varie en fonction de la préférence de chaque famille, et augmente si la famille est affectée à un jour différent de son premier choix. De plus, il y a une pénalité associée à la différence de population entre chaque jour consécutif de la semaine. L'objectif global est de minimiser la somme du coût d'affectation des familles et de la pénalité.

Nous disposons de 5 jeux de données en .csv qui comprennent chacun entre 400 et 500 familles. Certains jeux de données sont plus grands que les autres. Le premier par exemple, compte un peu plus de 1900 individus et le dernier compte un peu de moins de 1450 individus. Le sujet nous impose une limite maximal de 250 individus par jour néanmoins, nous allons rencontrer un problème avec le premier fichier .csv. En effet, 250 individus maximal multiplié par les 7 jours de la semaines font 1750 individus au maximum. Hors, le sujet nous impose de placer toutes les familles, soit les 1900 individus. Pour pallier à ce problème, on laissera le nombre maximal d'individu comme une variable à inscrire au préalable à notre fonction « Répartition » que l'on explicitera ci-après. On fera aussi attention à ne pas choisir un nombre maximal d'individu trop grand sous risque de rendre la contrainte du coût obsolète (en effet, mettre un trop grand nombre d'individu par jour impliquerait que tous les individus pourront choisir leur choix numéro 1 sans conflit).

Afin de répondre à ce problème, j'ai raisonné dans un premier temps avec le langage de programmation Python pour des questions pratiques (davantage de maîtrise du langage, plus facile pour résoudre des problèmes d'optimisation, plus d'expérience avec). Les étapes que j'ai suivi pour résoudre le problème sont les suivantes :

- Récupérer les jeux de données afin de pouvoir les exploiter, c'est à dire les mettre dans un format pratique pour mon code (récupérer une matrice de 6 colonnes par le nombre de famille présent dans le set)
- Programmer une fonction « Répartition » pour pouvoir calculer le coût et la pénalité. C'est le cœur du projet et il est question d'optimisation à la fois dans le code et dans la résolution du problème au travers de cette fonction. Pour ce faire :
    - En entrée de la fonction, on récupère la matrice de 6 par le nombre de famille présent dans le jeu de données ainsi que le nombre maximal d'individu.
    - On initialise une liste de taille 7 composée uniquement de 0 que l'on note J. Elle représente le nombre de personne présente par jour dans le parc.
    - Ensuite, on cherche à attribuer le plus de personnes à leur premier choix. Donc on fait une boucle sur toute la matrice d'entrée pour attribuer au maximum de personnes leur premier choix. Tant que le nombre maximal de personnes pour le choix numéro 1 de la famille n'est pas atteint, on donne à la famille en question sont choix numéro 1. Si le nombre maximal est atteint, on affecte la famille à une liste notée « deux » qui repassera par la même boucle mais pour le choix numéro 2.

    - On réitère l'opération 5 fois jusqu'à aboutir à tous les préférences de toutes les familles.
- Une fois la répartition effectué, en fin de fonction « Répartition », on récupère la liste J mais aussi la matrice initiale comportant toutes les familles et leur affectation mais à un détail près. Grâce à la fonction répartition, nous avons aussi indiqué dans la matrice initiale la préférence qui a été attribué à toute les familles (valeur qui varie de 1 à 5) ce qui nous facilitera la tâche pour l'écriture des fonctions « Coût » & « Pénalité »
- Les fonctions « Coût » & « Pénalité » se rédigent sans trop de difficulté, il suffit de reprendre la structure imposé par le sujet.

Le programme en Python est terminé. Il est question maintenant de l'implémenter en C. La fonction « Répartition » a été la plus délicate à ré-implémenter ainsi que de faire appel au fichier CSV dans un format pratique. La structure du code reste sensiblement identique néanmoins.

Les résultats obtenus sont **intéressants**. Pour le premier jeu de données pb10.csv, avec un nombre maximal d'individus de 280, on obtient un coût de 1731 et une pénalité de 135.3 soit une fonction objectif évalué en les conditions & contraintes ci-avant égale à 1865.3.
Cette limite d'individu a été dressé à 280 puisque le jeu de données pb10.csv compte 1929 individus et 280*7=1960. Il faut que la limite reste cohérente & proche du nombre de maximal d'individus.

Pour le 2nd jeu de données pb20.csv, on fixe la limite à 270 individus.
On obtient 721 de coût et 177.9 de pénalités soit 889.9 pour la fonction objectif.
Si l'on fixe la limite à 260, on obtient un coût bien plus élevés égale à 1414 et une pénalité égale à 108.8 soit une fonction objectif égale à 1522.8. On remarque un phénomène rassurant : plus la limite d'individus est fixé proche de la valeur du nombre de personnes présente dans toutes les familles, plus le coût est élevé.

Pour le 3ème jeu de données pb30.csv, on compte 1563 individus. Divisé par 7, on trouve 223,3. On arrondi à la l'entier supérieur pour fixer notre limite d'individus, soit 224. On trouve ainsi un coût de 1066 et une pénalité de 71.2 soit une fonction objectif qui a pour valeur 1137.2.

On résonne de la même manière pour le 4ème jeu de donnée pb40.csv. On fixe la limite à 213 pour obtenir un coût de 1747 et une pénalité de 58.1 soit une fonction objectif égale à 1803.1

Pour le dernier jeu de donnée pb50.csv, on montre le phénomène qui apparaît lorsque le nombre maximal d'individu est fixée beaucoup trop haut. On compte 1436 individu dans le jeu de données. En fixant la limite à 300, on obtient un coût de 0 mais une pénalité beaucoup plus grande que tous les autres essaie qui ici est égale à 604. La pénalité est beaucoup plus grande que dans tous les autres essaies puisque tous les individus ont obtenus leur choix numéro 1 et la différence de population entre les jours est beaucoup plus notables que dans les autres essaie. Au jour numéro 1, on compte par exemple 246 individus, pour seulement 174 au jour 2.